Teste de Mesa

APRESENTAÇÃO

Após a construção do algoritmo, realizamos um teste chamado teste de mesa para verificar se o algoritmo construído realiza mesmo a tarefa para a qual foi projetado. Existem várias formas de realizar o teste de mesa, seja utilizando uma ferramenta, seja de forma manual; algumas formas são mais detalhadas e outras, mais resumidas.

O que importa é que a validação de um algoritmo é essencial para a sua qualidade.

Nesta Unidade de Aprendizagem, estudaremos a técnica de teste de mesa, como construir e realizar esse tipo de teste em fluxograma e pseudocódigo.

Bons estudos.

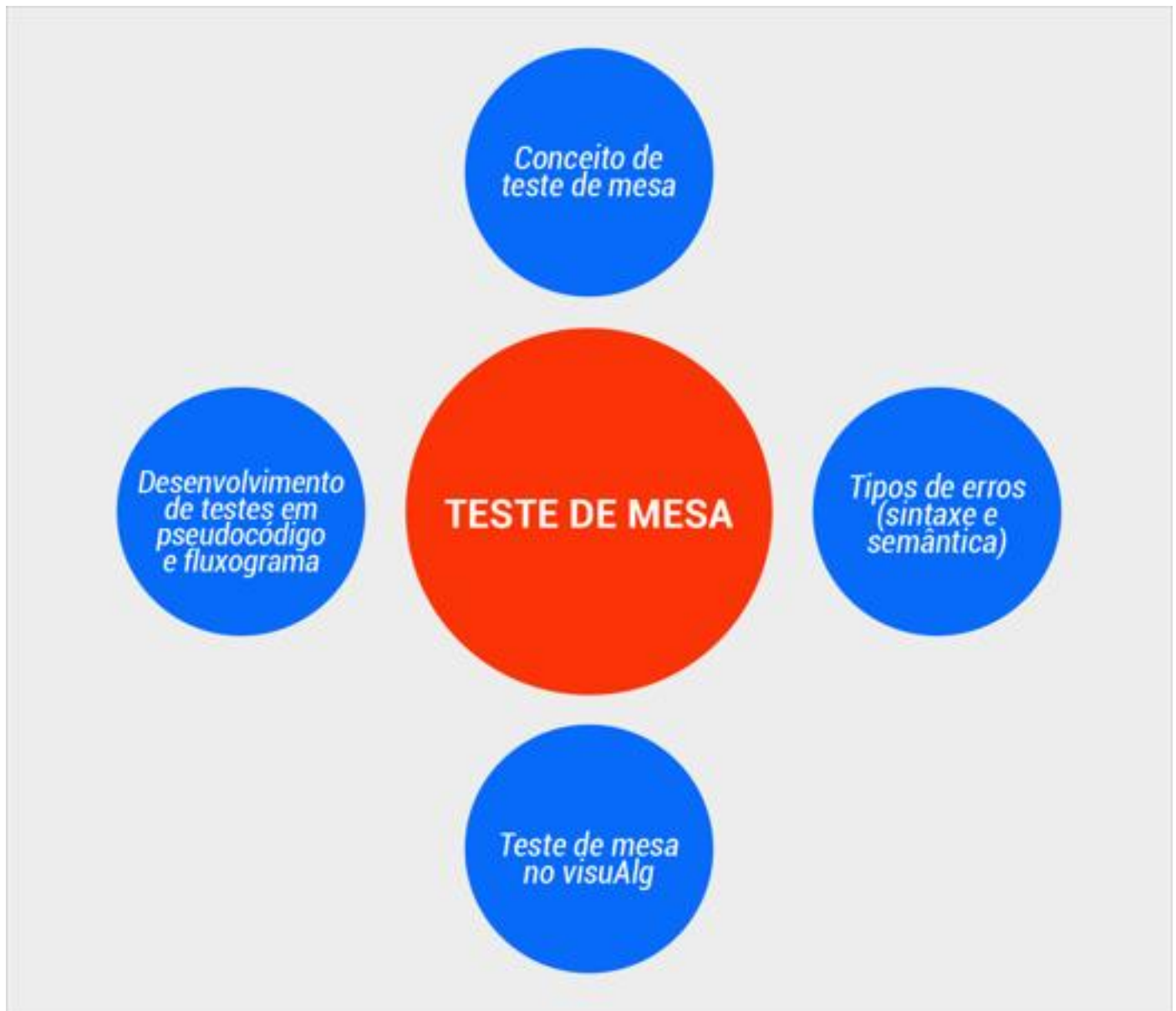
Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Reconhecer a importância de um teste de mesa.
- Usar o teste de mesa em algoritmos na forma de pseudocódigo e fluxograma.
- Contrastar os tipos de erros de sintaxe e semântica.

INFOGRÁFICO

Após a construção do algoritmo, realizamos um teste chamado teste de mesa para verificar se o algoritmo construído realiza mesmo a tarefa para a qual foi projetado.

Existem várias formas de realizar o teste de mesa, seja utilizando uma ferramenta, seja de forma manual; algumas formas são mais detalhadas e outras, mais resumidas. O que importa é que a validação de um algoritmo é essencial para a sua qualidade.

Nesta Unidade de Aprendizagem, estudaremos a técnica de teste de mesa, como construir e realizar esse tipo de teste em fluxograma e pseudocódigo.

CONTEÚDO DO LIVRO

Através do teste de mesa, conseguimos verificar a eficácia (corretude) de determinado algoritmo.

Para isso, precisamos testar com um conjunto variado de dados de entrada, abrangendo a maior quantidade possível de situações que poderão ocorrer durante a utilização do algoritmo.

Para compreender melhor o objetivo de um teste de mesa, acompanhe um trecho da seguinte obra: EDELWEISS, N.; LIVI, M.A.C. **Algoritmos e programação com exemplos em Pascal e C -** Vol. 23. Série Livros Didáticos Informática UFRGS. Porto Alegre: Bookman, 2014.

23

■■ série livros didáticos informática ufrgs ■■

algoritmos e programação

com exemplos em Pascal e C

▪▪ nina edelweiss

▪▪ maria aparecida castro livi

as autoras

Nina Edelweiss é engenheira eletricista e doutora em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Durante muitos anos, lecionou em cursos de Engenharia e de Ciência da Computação na UFRGS, na UFSC e na PUCRS. Foi, ainda, orientadora do Programa de Pós-Graduação em Ciência da Computação da UFRGS. É coautora de três livros, tendo publicado diversos artigos em periódicos e em anais de congressos nacionais e internacionais. Participou de diversos projetos de pesquisa financiados por agências de fomento como CNPq e FAPERGS, desenvolvendo pesquisas nas áreas de bancos de dados e desenvolvimento de software.

Maria Aparecida Castro Livi é licenciada e bacharel em Letras, e mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Desenvolveu sua carreira profissional na UFRGS, onde foi programadora e analista de sistema, antes de ingressar na carreira docente. Ministrou por vários anos a disciplina de Algoritmos e Programação para alunos dos cursos de Engenharia da Computação e Ciência da Computação. Sua área de interesse prioritário é o ensino de Linguagens de Programação, tanto de forma presencial quanto a distância.

E22a Edelweiss, Nina.

Algoritmos e programação com exemplos em Pascal e C [recurso eletrônico] / Nina Edelweiss, Maria Aparecida Castro Livi. – Dados eletrônicos. – Porto Alegre : Bookman, 2014.

Editado também como livro impresso em 2014.
ISBN 978-85-8260-190-7

1. Informática. 2. Algoritmos – Programação. I. Livi, Maria Aparecida Castro. II. Título.

CDU 004.421

Catalogação na publicação: Ana Paula M. Magnus – CRB 10/2052

internamente em computadores é o binário (base 2), as capacidades são representadas como potências de 2:

K	1.024	2^{10}
M	1.048.576	2^{20}
		etc...

A grafia dos valores expressos em múltiplos de *bytes* pode variar. Assim, por exemplo, 512 quilobytes podem ser escritos como 512K, 512KB, 512kB ou 512Kb. Já os valores expressos em bits, via de regra, são escritos por extenso, como em 512 quilobits.

1.6 dicas

Critérios que devem ser observados ao construir um algoritmo:

- procurar soluções simples para proporcionar clareza e facilidade de entendimento do algoritmo;
- construir o algoritmo através de refinamentos sucessivos;
- seguir todas as etapas necessárias para a construção de um algoritmo de qualidade;
- identificar o algoritmo, definindo sempre um nome para ele no cabeçalho. Este nome deve traduzir, de forma concisa, seu objetivo. Por exemplo: `Algoritmo 1.1 – Soma2` indica, através do nome, que será feita a soma de dois valores;
- definir, também no cabeçalho, o objetivo do algoritmo, suas entradas e suas saídas;
- nunca utilizar desvios incondicionais, como `GOTO` (`VÁ PARA`).

1.7 testes

Testes de mesa. É importante efetuar, sempre que possível, testes de mesa para verificar a eficácia (corretude) de um algoritmo antes de implementá-lo em uma linguagem de programação. Nestes testes, deve-se utilizar diferentes conjuntos de dados de entrada, procurando usar dados que cubram a maior quantidade possível de situações que poderão ocorrer durante a utilização do algoritmo. Quando o algoritmo deve funcionar apenas para um intervalo definido de valores, é recomendável que se simule a execução para valores válidos, valores limítrofes válidos e inválidos e valores inválidos acima e abaixo do limite estabelecido. Por exemplo, se um determinado algoritmo deve funcionar para valores inteiros, no intervalo de 1 a 10, inclusive, o teste de mesa deveria incluir a simulação da execução para, no mínimo, os valores 0, 1, 10, 11 e um valor negativo.

- no cabeçalho, descrevendo qual a finalidade do programa;
- junto às declarações das variáveis, explicando o que cada variável vai armazenar, a menos que os nomes sejam autoexplicativos;
- junto aos principais comandos.

incremento/decremento de variáveis em C. Evitar misturar, em um mesmo código e para uma mesma variável, as duas formas de incremento/decremento de variáveis apresentadas.

revisar os formatos utilizados. Nas funções de C que utilizam formatos, revisá-los com atenção. Formatos incorretos podem gerar erros difíceis de serem detectados.

3.10 testes

incluir comandos de saída para depurar programas. Uma forma de depurar um programa é usar diversos comandos de saída ao longo do programa para acompanhar os valores que são assumidos por algumas das variáveis durante o processamento. Uma vez feitos todos os testes necessários, esses comandos devem ser retirados do programa.

Por exemplo, no Algoritmo 3.1 poderia ser acrescentado um comando de saída logo após a leitura, para verificar se os valores lidos são mesmo aqueles que foram fornecidos. Para facilitar a remoção desses comandos auxiliares do programa, sugere-se que sejam alinhados de forma diferente dos demais comandos:

```
Algoritmo 3.1 – Soma2
  {INFORMA A SOMA DE DOIS VALORES LIDOS}
  Entradas: valor1, valor2 (real){VALORES LIDOS}
  Saídas:   soma (real)
início
  ler (valor1, valor2)            {ENTRADA DOS 2 VALORES}
  escrever (valor1, valor2)       {PARA TESTE}
  soma ← valor1 + valor2          {CALCULA A SOMA}
  escrever (soma)                 {INFORMA A SOMA}
fim
```

testar para todos os dados de entrada possíveis. Outro aspecto importante para garantir resultados corretos é realizar testes com todas as possíveis combinações de dados de entrada, incluindo valores positivos, negativos e nulos. Testar também o que acontece quando são fornecidos tipos de dados incorretos na entrada. Na Tabela 3.7, são mostrados alguns pares de valores de entrada que poderiam ser utilizados para testar o Algoritmo 3.1 (adaptando os números à sua representação na linguagem de programação utilizada):

tabela 3.7 Exemplos de valores de entrada a serem testados

valor1	valor2	
0	0	dois valores nulos
0	5	primeiro valor nulo
-2	0	segundo valor nulo
-3	0	um valor negativo e um nulo
0	2.3	um valor real e um nulo
5	5	dois valores iguais
2	5	dois valores inteiros diferentes
3.5	2.7	dois valores reais diferentes
100000	-3.7	um positivo e um negativo
-7	-4.67	dois valores negativos

3.11 ⇢ exercícios sugeridos

exercício 3.1 Escreva uma expressão lógica que seja verdadeira no caso do valor contido em uma variável inteira `valor` estar compreendido entre os valores 10 e 50, incluindo os limites.

exercício 3.2 Leia as coordenadas de dois pontos no plano cartesiano e imprima a distância entre esses dois pontos. Fórmula da distância entre dois pontos ($x1$, $y1$) e ($x2$, $y2$):

$$\sqrt{(x2-x1)^2+(y2-y1)^2}$$

exercício 3.3 Dados três valores armazenados nas variáveis a, b e c, calcule e imprima as médias aritmética, geométrica e harmônica desses valores. Calcule também a média ponderada, considerando peso 1 para o primeiro valor, peso 2 para o segundo e peso 3 para o terceiro.

Fórmulas: média aritmética: $\frac{a+b+c}{3}$

média geométrica: $\sqrt[3]{a \times b \times c}$

média harmônica: $\frac{3}{\frac{1}{a}+\frac{1}{b}+\frac{1}{c}}$

média ponderada: $\frac{1a+2b+3c}{1+2+3}$

usar comandos aninhados em sequências de testes. Quando vários testes relacionados tiverem que ser feitos, utilizar preferencialmente comandos de seleção dupla aninhados, em vez de sequências de comandos de seleção simples.

usar seleção múltipla em lugar de sequências de seleções simples. Para maior clareza, utilizar um comando de seleção múltipla em vez de sequências de comandos de seleção simples sempre que a linguagem oferecer essa construção.

não repetir desnecessariamente testes semelhantes. Para testes mutuamente exclusivos, utilizar comandos de seleção dupla, evitando assim a repetição desnecessária de testes.

planejar os testes cuidadosamente. Planejar bem os testes para verificar o maior número possível de casos diferentes. Quanto mais testes bem planejados forem realizados, maior a garantia de que o programa será executado corretamente.

4.10 testes

testar o maior número possível de caminhos de execução. Diversos testes devem ser realizados para testar a corretude de um trecho de programa que contém um comando de seleção. Devem ser criados conjuntos de dados que façam a execução do programa passar pelo maior número de caminhos possíveis. Tanto comandos de seleção simples quanto de seleção dupla devem ser testados com pelo menos dois conjuntos de dados, um que faça a expressão lógica resultar verdadeira e outro que a torne falsa. Por exemplo, o comando:

```
se média ≥ 6
então escrever('Aprovado')
```

deve ser testado com dados que resultem em (1) média maior que 6, (2) média igual a 6 e (3) média menor que 6. Os mesmos conjuntos de dados podem ser utilizados para testar o comando:

```
se média ≥ 6
então escrever('Aprovado')
senão escrever('Reprovado')
```

verificando se as mensagens corretas são apresentadas para os vários conjuntos de dados utilizados.

Um cuidado especial deve ser tomado para criar os conjuntos de dados no caso de comandos aninhados a fim de que o maior número possível de caminhos seja testado. Por exemplo, no comando:

```
se média ≥ 9
então conceito ← 'A'
senão se média ≥ 7,5
      então conceito ← 'B'
      senão se média ≥ 6,0
```

```
então conceito ← 'C'
senão conceito ← 'D'  {MÉDIA < 6}
```

devem ser criados dados para testar cada um dos intervalos de média, incluindo os limites de cada um deles. Por exemplo, testar com valores que resultem nas seguintes médias: 10,0 – 9,5 – 9,0 – 8,0 – 7,5 – 7,0 – 6,0 – 4,0 – 0,0.

testar com dados incorretos. Outro cuidado a ser tomado é fazer testes com valores de dados incorretos para ver como o programa se comporta nesses casos. A execução do programa não deve parar quando forem fornecidos dados incorretos. Quando isso acontecer, o programa deve informar ao usuário a ocorrência de erro. Por exemplo, no código anteriormente citado, caso as notas fornecidas resultem em uma média maior do que 10, o programa atribuirá ao aluno o conceito "A", o que não estará correto. Caso a média resulte em valor menor do que zero, o conceito atribuído será "D", o que também estará incorreto. Se o programa, antes de entrar nesse comando, não fizer um teste da corretude dos dados fornecidos, esse trecho de programa estará sujeito a erro.

testar o pior e o melhor caso. Além dos casos médios, sempre testar o pior e o melhor caso, de acordo com os dados fornecidos.

4.11 ··→ exercícios sugeridos

exercício 4.1 Faça um programa que leia dois valores, o primeiro servindo de indicador de operação e o segundo correspondendo ao raio de um círculo. Caso o primeiro valor lido seja igual a 1, calcular e imprimir a área desse círculo. Se o valor lido for 2, calcular e imprimir o perímetro do círculo. Se o valor lido for diferente desses dois valores, imprimir uma mensagem dizendo que foi fornecido um indicador incorreto para a operação a ser realizada.

exercício 4.2 Leia três valores e informe se podem corresponder aos lados de um triângulo. Em caso afirmativo, verifique e informe se esse triângulo é:

- **a** equilátero;
- **b** isósceles;
- **c** escaleno;
- **d** retângulo.

exercício 4.3 Leia três valores e armazene-os nas variáveis A, B e C. Se todos forem positivos, calcule e imprima a área do trapézio que tem A e B por bases e C por altura.

exercício 4.4 Escreva um programa que calcule as seguintes conversões entre sistemas de medida:

- **a** dada uma temperatura na escala Celsius, fornecer a temperatura equivalente em graus Fahrenheit e vice-versa (Fórmula de conversão: 1º F = (9 / 5)º C + 32);
- **b** dada uma medida em polegadas, fornecer a equivalente em milímetros e vice-versa (Fórmula de conversão: 1 pol = 24,5 mm).

DICA DO PROFESSOR

O teste de mesa é muito importante, principalmente para os programadores iniciantes, pois é o meio pelo qual podemos acompanhar a execução de um algoritmo, passo a passo. Podemos encontrar erros e confirmar se a lógica utilizada está correta.

Muitos são os modelos de teste de mesa, alguns mais simples, outros mais complexos.

Assista ao vídeo para conhecer um pouco mais sobre os testes de mesa e sua construção, além de compreender e analisar algumas soluções práticas apresentadas de aplicação de teste de mesa em pseudocódigo e em fluxograma.

Conteúdo interativo disponível na plataforma de ensino!

EXERCÍCIOS

1) **Considere os seguintes valores iniciais para as variáveis: a = 3, b = 2, c = 2 e d = 4. Realize o teste de mesa e obtenha o valor da variável inteira x dos trechos de programa a seguir.**

I	II	III
x <- 0 se (a =3) ou (b < 2) entao x = (a /2) * quad(d)) fimse	x<- 1 se (não(a >= 2) ou não(b < 6)) entao x = c + d\b fimse x <- x + 2	x <-2 se (a + 2 <> b+2) e não(a>b) entao x <- raizq(d) + exp(a, b) x <- x + a mod d fimse x <- 0

São corretos os valores das alternativas I, II e III, respectivamente, em:

A) 24, 3 e 0.

B) 24, 0 e 0.

C) 24, 6 e 0.

D) 24, 4 e 0.

E) 24, 4 e 11.

2) **A aplicação do teste de mesa nos permite validar o algoritmo desenvolvido e, conforme os erros que vão se apresentando, podemos corrigir e testar novamente com novos valores. Considerando o conceito e as características do teste de mesa , assim como os tipos de erros de sintaxe e semântica encontrados nos algoritmos, analise as sentenças a seguir. I – O _____________ tem por objetivo verificar a corretude do algoritmo.**

II – O erro de __________ representa erros de escrita de comandos.

III – O erro de _____________ pode exibir comportamentos inadequados dos programas.

IV – O erro de __________ impede a execução do algoritmo.

V – O erro de __________ representa erros lógicos.

Complete as sentenças conforme as alternativas apresentadas abaixo. É correta a ordem apresentada apenas em:

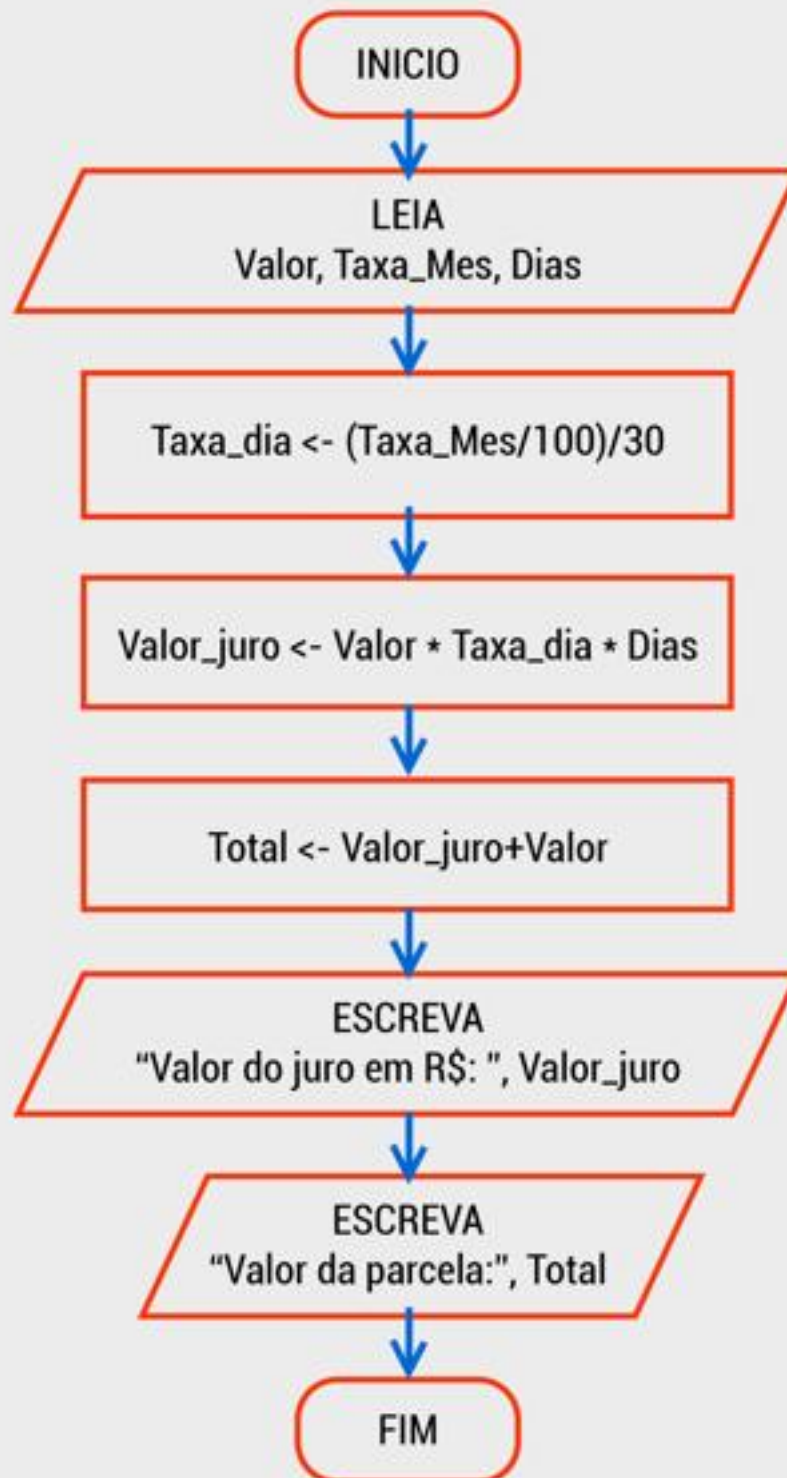
A) Teste de mesa – semântica – sintaxe – sintaxe - semântica.

B) Teste de mesa – sintaxe – semântica – sintaxe - semântica.

C) Teste de mesa – semântica – semântica – semântica - sintaxe.

D) Teste de mesa – sintaxe – sintaxe – semântica - semântica.

E) Teste de mesa – sintaxe – sintaxe – sintaxe - semântica.

3) Observe o algoritmo em fluxograma.

Realize o teste de mesa com os seguintes valores de entrada:

Valor = 2.500,00

Taxa_mes = 22%

Dias = 12

O valor total será?

A) 220,00

B) 9.100,00

C) 2.280,00

D) 2.720,00

E) 2.500,00

4) Dado o seguinte algoritmo.

```
Algoritmo "exercício_teste"
Var
   i,m : inteiro
   a: real

inicio
         leia(m)
         se (m<> 0) entao
                  i <- m \ 5
                  a<- m/12
         fimse
         se (m mod 2 >= 6) entao
                i <- i + 1
         se (m mod 2 <6) entao
                i <-i − 1
         fimse
         Escreva (a, i)
fimalgoritmo.
```

Realize o teste de mesa e identifique que valores serão escritos como saída para as variáveis a e i sendo lido 12 para a variável m?

A) a = 0 e i = 2.

B) a = 1 e i = 1.4.

C) a = 1 e i = 2.

D) a = 0 e i = -1.

E) a = 1 e i = 1.

5) **Dado o algoritmo.**

```
1 Algoritmo "teste"
2 var
3           l, area : real
4 Inicio
5   Escreval("Digite o lado do quadrado:")
6   Leia(l)
7   area <- l*l
8   Escreva("A área do quadrado é = ", area)
9 fimalgoritmo
```

Analise as alternativas de construção do teste de mesa para calcular a área de um quadrado.

Alternativa 1

Em execução	Variáveis		Saída
Nº Linha	l	area	
5			Digite o lado do quadrado:
6	[4]		
7		16	
8		{16}	A Área do quadrado é = 16

Alternativa 2

Em execução	Variáveis		Saída
Nº Linha	I	área	
5			Digite o lado do quadrado:
6	[4]		
7	4	16	
8	4	{16}	A Área do quadrado é = 16

Alternativa 3

Variáveis	
I	área
4	
	16

É correto o teste de mesa representado em:

A) I.

B) I e II.

C) I e III.

D) II e III.

E) Todas as alternativas.

Cálculo do frete (sobre o peso) - Correios

Um das grandes dúvidas dos consumidores que adquirem produtos online é a respeito do cálculo do frete de uma encomenda. De acordo com informações do correio, o frete é elaborado a partir de vários pontos, como preço, tamanho do produto e distância a ser percorrida.

Primeiramente, precisamos calcular o peso cúbico dos produtos; para isso, utiliza-se a fórmula:

C x L x A / 6.000

Em que:

C – representa comprimento

L – representa largura

A – representa altura

O resultado dessa equação é o peso cúbico da entrega.

Exemplo: Se um produto tiver 10 kg ou menos, considera-se o seu peso físico, e não o peso cúbico.

Produto = 5,2 kg de peso físico

Com as seguintes dimensões: 50 cm de comprimento; 40 cm de largura e 20 cm de altura, o peso cúbico é 3.33 kg.

Peso = (50 x 40 x 20 / 6.000) = 3.33 kg

Nas encomendas com peso físico superior a 10 kg, conta o maior resultado na comparação entre ambos os pesos (cúbico e físico).

Exemplo:

Se uma encomenda pesa 15 kg e possui as seguintes medidas:

43 cm de comprimento; 28 cm de largura e 52 cm de largura:

Peso = (43 x 28 x 52 / 6.000) = 10,4 kg

Nesse caso, para efeitos de custo de transporte, será considerado o peso físico da encomenda: 15 kg.

Agora vamos desenvolver um algoritmo em pseudocódigo para realizar o cálculo que o correio utiliza para cobrar o frete. Como saída, deverá imprimir na tela o peso físico e o peso cúbico e qual dos pesos será utilizado para o calculo do frete.

A representação para o cálculo do peso do frete ficaria:

```
1  Algoritmo "peso_frete"
2 Var
3    peso_fisico, peso_cubico, c,l,a, cobrado_peso: real

4 inicio
5    Escreval("Digite o peso do produto em Kg:")
6    Leia(peso_fisico)
7    Escreval("Digite o comprimento:")
8    Leia(c)
9    Escreval("Digite a largura:")
10    Leia(l)
11    Escreval("Digite a altura:")
12    Leia(a)
13    peso_cubico <- c*l*a/6000
14    Escreval("Peso Cúbico = ", peso_cubico)
15    se (peso_fisico <=10) entao
16          cobrado_peso <- peso_fisico
17    fimse
18    se (peso_fisico  >10) entao
19         se peso_cubico > peso_fisico entao
20              cobrado_peso <- peso_cubico
21         fimse
22         se peso_fisico>= peso_cubico entao
23               cobrado_peso <- peso_fisico
24         fimse

25     fimse
26     Escreval("O peso que será cobrado será de : ", cobrado_peso)
27 fimalgoritmo
```

Vamos realizar o teste de mesa?

Temos de fazer dois testes, um com peso físico > que 10 kg e outro com <= de 10 kg para atender as duas situações de cálculo do peso.

Valores para teste:

Valores com peso físico acima e abaixo de 10.

Com pesos físicos acima de 10 kg tem duas situações: com peso físico maior que peso cúbico e outra de peso cúbico maior que peso físico.

Teste de mesa: situação com peso físico menor que 10:

Em execução	Variáveis						Saída
Nº linha	Peso_fisico	c	l	a	Peso_cubico	Cobrado_peso	
5							Digite o peso físico do produto:
6	9						
7							Digite o comprimento:
8		60					
9							Digite a largura:
10			20				
11							Digite a altura:
12				40			
13					8		
14					{8}		Peso cúbico = 8
16						9	
26						{9}	O peso que será cobrado será de: 9

Teste de mesa: situação com peso físico maior que 10 (peso cúbico >peso físico):

Em execução	Variáveis						Saída
Nº linha	Peso_fisico	c	l	a	Peso_cubico	Cobrado_peso	
5							Digite o peso físico do produto:
6	12						
7							Digite o comprimento:
8		90					
9							Digite a largura:
10			50				
11							Digite a altura:
12				80			
13					60		
14					{60}		Peso cúbico = 60
16						60	
26						{60}	O peso que será cobrado será de: 60

Teste de mesa: situação com peso físico maior que 10 (peso cúbico <= peso físico):

Em execução	Variáveis						Saída
Nº linha	Peso_fisico	c	l	a	Peso_cubico	Cobrado_peso	
5							Digite o peso físico do produto:
6	15						
7							Digite o comprimento:
8		30					
9							Digite a largura:
10			10				
11							Digite a altura:
12				20			
13					1		
14					{1}		Peso Cúbico = 1
16						15	
26						{15}	O peso que será cobrado será de : 15

Agora, testamos todas as possibilidades e o algoritmo está funcionando.

Outra forma simplificada de teste de mesa poderia ser desenvolvida em uma única tabela:

Teste de mesa: situação com peso físico maior que 10 (peso cúbico > peso físico):

	Variáveis					
testes	Peso_fisico	c	l	a	Peso_cubico	Cobrado_peso
Teste1	9	60	20	40	8	9
Teste2	12	90	50	80	60	60
Teste3	15	30	10	20	1	15

SAIBA MAIS

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Como fazer teste de mesa?

Conteúdo interativo disponível na plataforma de ensino!

Questões com teste de mesa

Conteúdo interativo disponível na plataforma de ensino!